Simpli

Simpli Integration Manager Configuration Guide

Version Control

Version	Date	Notes	Author
1.0	1/4/2020	Document Creation	Tom Ansley

Table of Contents

Table of Tables

Table of Figures

Introduction

Integrations between systems can be complex for several reasons. Lots of data gets passed around unnecessarily, there are often redundant services and system specific metadata seeps into integrations unnecessarily. These complexities create large amounts of system debt which manifests itself over time. Coupling between systems increase as integrations are added. The back and forth conversations between systems becomes more complex.

The Simpli Integration Manager (Simpli) reduces complexity and removes coupling between integrated systems and Salesforce. How are these problems solved?

- Integrations are defined via configuration using standard Salesforce point and click.
- A single, simple integration interface is used for all inbound requests.
- All inbound fields are mapped through configuration.
- Integrations can be flexibly configured to handle only what is needed for processing.
- Processing is service oriented rather than table driven.

There are many other features also available including -

- SOAP and REST interfaces available as well as ability to use internally through Apex.
- A Lightning console to easily view descriptions of all configured processes.
- A process test page is available to quickly test configurations.
- Can be extended based on specific user needs through development
- An ability to have a process run against multiple records in batch.
- Can be used for automated testing.

Please note the following - **Simpli is best suited to inbound transactional and micro-service processing. Although there is a batch component it is not designed for large-scale data loads.**

Configuration Basics

Once the application has been downloaded and installed an app is made available called Simpli Integration Manager.

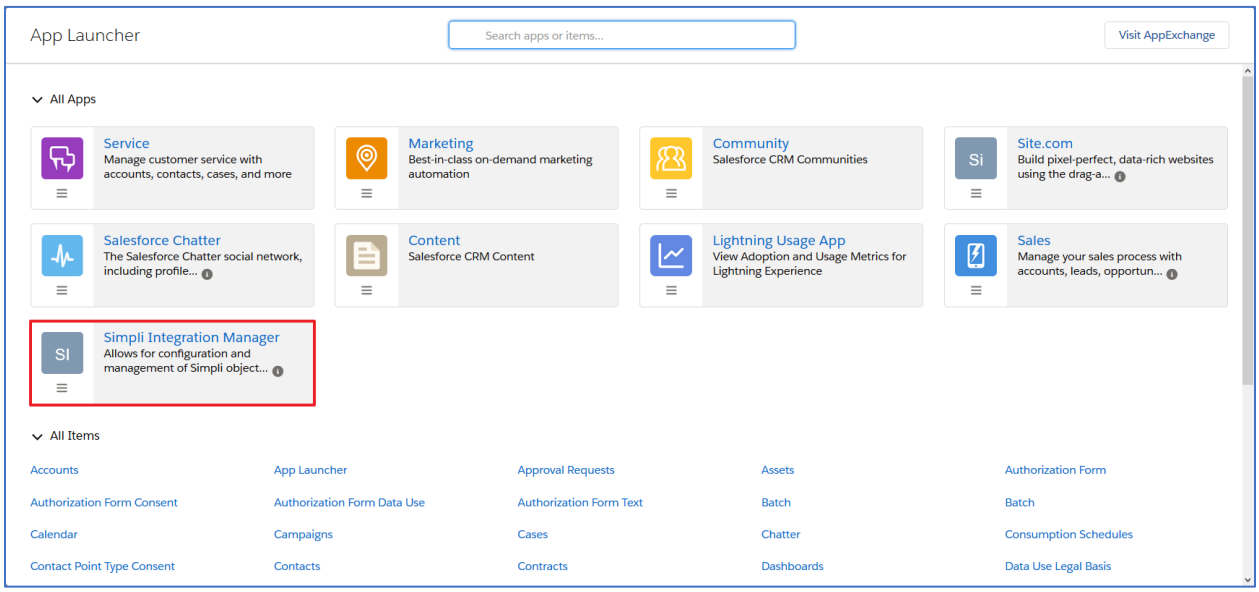


Figure 1 - Simpli App Screenshot

Click on the app. After the app opens, a number of tabs are made available. The tabs we will be interested in for initial configuration are –

- **Simpli Objects** –
- **Simpli Attributes** –

The first four tabs (Start, Describe, Process, Batch) assume that a configuration has already been created and are used for viewing, describing and processing the configurations. There are two main areas of configuration within a Simpli object –

1. Configuration of the object itself along with all child object configuration. i.e. opportunity, opportunity line item etc.
2. The attributes of the object. i.e. initial value, requiredness, overridability, external name etc.

Before creating a configuration, it is always a good idea to draw out the relationships that exist between all the objects. This will help identify what objects need to be created along with where in the object graph they lie. Once the objects have been identified the attributes of these objects can be added, with careful attention being paid to whether they are to be set by the user or defaulted by the configuration.

Simpli Objects

The Simpli Object is a Salesforce custom object. It describes, from an object perspective, how to process the object. i.e. what Salesforce object the configuration relates to, whether it has a parent and what Simpli logic should be used when processing the object. Below is a table describing the fields of the Simpli Object.

Field Name	Description
Simpli Object Name	A unique name which identifies the configuration. This is the name used when calling the Simpli framework and asking to process a configuration.
Class Name	The Apex class which holds the processing logic which will be used when processing the object. There are standard Apex classes which are available (described below). Custom classes can also be created and used.
SFDC Object API Name	Holds the SFDC Object API name that represents this Simpli object. This is not a required field as the framework allows for processing where there is no SFDC object. See ***Example - Send Single Email*** for a configuration that does not use an SFDC object.
Parent Object	This is used when a hierarchy of processing needs to be performed. This is useful for when parent/child relationships need to be created. See ***Example - Opportunity Insert*** for a configuration that has child relationships.
Is Active	Identifies if the object is active. If it is not, then any application call to use the configuration will fail.
Allow Multiples	This indicates whether multiples objects of this kind can be created. This is useful when an unknown number of children objects of the same type need to be processed. See ***Example - Opportunity Line Items Insert (Multi)*** for an example illustrating this behavior. Note this cannot be set to true when this object has no parent.
Description	This is used by the Describe tab when viewing a description of the configuration. It is useful to describe the configuration here. As the number of configurations increase it helps identify particular configuration logic

Table 1 – Simpli Object SFDC Field Descriptions

Simpli Attributes

The Simpli Attribute describes the data that should be used when processing the Simpli Object. i.e. the action that should be performed (find, insert, update, publish, clone), any field data that should be populated etc. The following are different ways attributes can be configured –

1. **Formula** - as formulas where the data for the field is determined based on the formula.
2. **Lookup** - as lookup values where the data for the field is used to find associated objects which are used to populate the attributes parent object.
3. **Data** - as data fields where the data for the field is passed in when handling CRUD functionality within SFDC.
4. **Config** - as Simpli config fields where the data determines how the Simpli object is processed. This data is not stored directly in SFDC.

Each of these kinds of attributes will be discussed in more detail below. For the moment, here is a table describing the fields of the Simpli Attribute.

Field Name	Description

Allow Multiples	Indicates whether the attribute can be supplied in multiples i.e. many values for one attribute. This is typically used when the attribute is used for an indeterminate list of objects. i.e. Opportunity Line Items.
Allow Overrides	Indicates whether the attribute can be overridden or not.
External Index	Used to identify in what order attributes that are available externally should be displayed to the user.
External Name	The name of the attribute supplied by a calling application. This is the only way an external application can refer to the attribute.
Formula	Holds the formula which indicates where data is to be retrieved from.
Is Formula	Identifies whether this attribute is a formula.
Is Lookup	Identifies whether this attribute is a lookup.
Is Required	Indicates whether this field must have a value for the configuration to process.
Lookup Field	Holds the object and field which identifies the lookup record to be returned. This is a required field if the attribute is a lookup attribute.
Lookup Id Value	The value compared against the above lookup field to find the lookup record. This is not a required field. The value will only be used if there is no value in the Value field. Note that this field can contain a formula to get the value from another attribute.
Lookup Return Field	The field on the lookup record that is to be returned. This is a required field if this attribute is a lookup attribute.
Parent Object	The Simpli Object that this attribute is a part of. Each attribute must have a parent object.
Name	The name of the attribute which is unique to the Simpli Object. This name is used internally by the framework for processing.
Type	Indicates whether this attribute is an input or output attribute. Only input attributes are used for processing. Output values are made available so that data can be returned as needed.
Value	The default value for this attribute.

Table 2 - Simpli Attribute SFDC Field Descriptions

As we start to go through examples the above field descriptions will start to make more sense.

Basic Example – Lead Insert

A simple example which is provided when installing the app from the AppExchange. NOTE – all the examples discussed in this document are provided with the app. The example is ***Example - Lead Insert***

This configuration has a single object which represents a lead. The fields of interest that have been configured are as follows –

Field Name	Field Value	Notes
Name	Example - Lead Insert	The name used when calling for the configuration to be processed.
Class Name	SimpliProcessSFDCObject	The base Simpli Apex class which represents an SFDC object, either standard or custom.
SFDC Object API Name	Lead	The API name of the object we want to represent.
Is Active	true	Indicating we want to process the configuration.

Table 3- Example - Lead Insert Object Fields

Nothing fancy going on here. We are creating a configuration, giving it a name and saying that we want it to represent a Lead object. But, by itself there is nothing to do. No kind of processing has been indicated. There is nothing in the above fields which identify what should happen to the lead. That is what the attributes contain. Here are the attributes.

Attr. Name	External Name	Type	Value	Is Required	Allow Override
ActionType		Input	Insert	true	false
Status		Input	Working - Contacted	true	false
FirstName	First Name	Input		true	true
LastName	Last Name	Input		true	true
Company	Company Name	Input		true	true
MobilePhone	Mobile Phone	Input		false	true
Email	Email	Input		true	true
LeadSource	Source	Input		false	true
Description	Description	Input		false	true

Table 4 - Example - Lead Insert Attribute Fields

Note the following –

1. **ActionType** - specifies the action that is being performed against the Lead object. In this case the lead will be inserted.
2. **Status** – an attribute name that corresponds to the API name on the Lead object. It has a value and cannot be overridden. This is a configuration choice which in this case has specified that all new leads must have their status default to “Working – Contacted” on insertion.
3. **Status** – notice also that when an attribute cannot be overridden there is no external name as the value will not be presented to the user for potential update.
4. **FirstName, LastName, Company, Email** – these are attribute names that correspond to the API names on the Lead. All fields are required and need to be set by the user. Because of this an external name is required. The external name is what is passed in by the calling application when supplying the value.
5. **MobilePhone, LeadSource, Description** – these attribute names also correspond to the API names on the Lead. These fields are not required but are available to be overridden hence them having external names.

We now have a configuration that is complete enough to process. Let’s process the configuration using the Process tab in the Simpli app. When we select the process, the following is displayed –

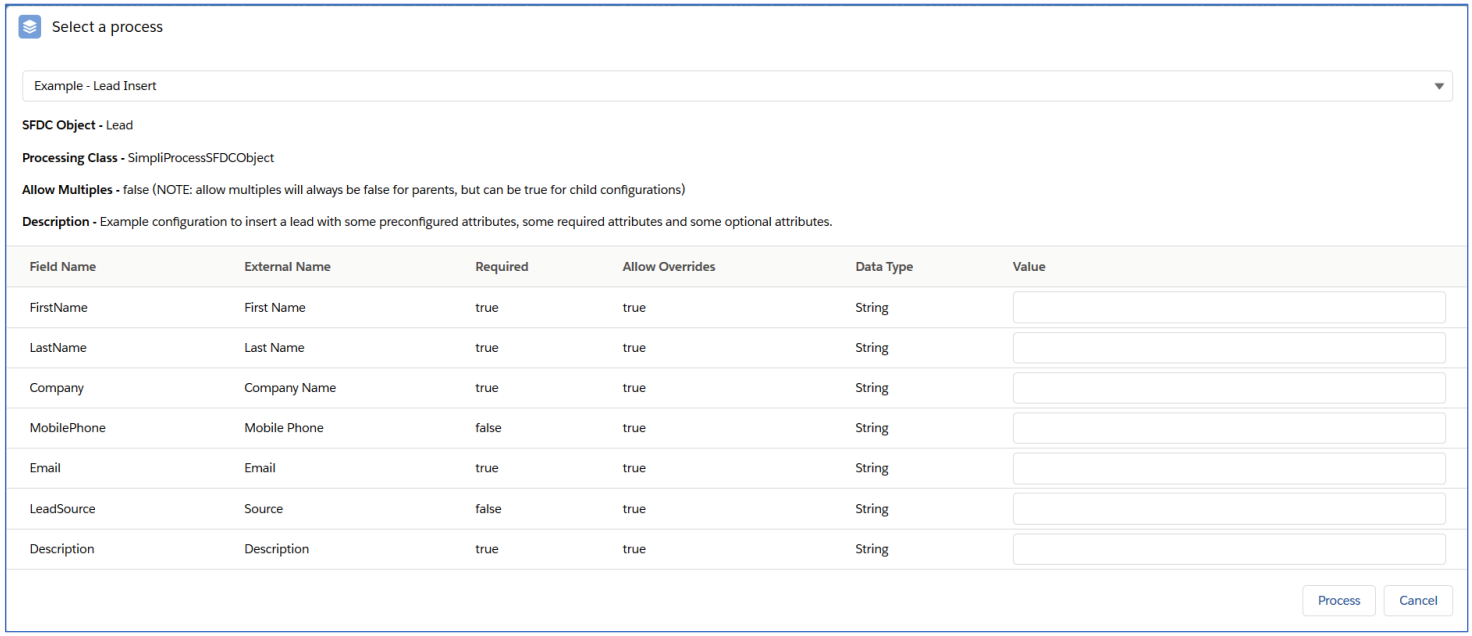


Figure 2 - Lead Insert Process Screenshot

Notice that only those fields the user can update are displayed. This is a design decision on this particular page. It helps show how although there may be a lot of data required in order to process a configuration the interface with the user can remain simple. (It is probable that apps using Simpli will only display the External Name and Value.)

If we enter the following values into the form –

External Name	Value
First Name	John
Last Name	Smith
Company Name	Simpli LLC
Mobile Phone	3031231234
Email	john@simplillc.com
Source	Email
Description	This is our first example!

and click Process we see the returned result.

Processing result (note that below we are displaying all output attributes)

Object Name	Field Name	External Name	Data Type	Value
Example - Lead Insert	SimpliResultGroupKey		String	
Example - Lead Insert	SimpliCreateResult		Boolean	false
Example - Lead Insert	SimpliClassName		String	SimpliProcessSFDCObject
Example - Lead Insert	SimpliDescription		String	Example configuration to insert a lead w
Example - Lead Insert	SFDCObject		Sobject	
Example - Lead Insert	SFDCObjectName		String	Lead
Example - Lead Insert	SFDCPlatformEventResult		Boolean	
Example - Lead Insert	SFDCPlatformEventErrorDesc		String	
Example - Lead Insert	Id		Id	00Q1C00001MGyStUAL
Example - Lead Insert	IsDeleted		Boolean	false
Example - Lead Insert	MasterRecordId		String	
Example - Lead Insert	LastName		String	Smith

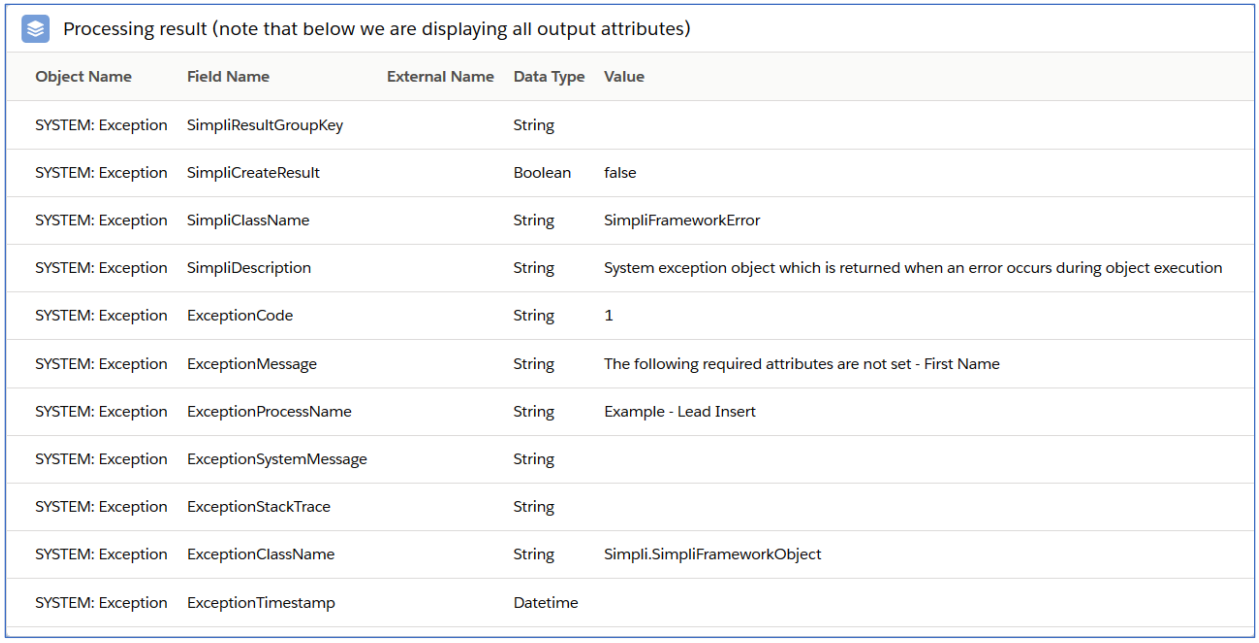
Figure 3 - Lead Insert Processing Reult Screenshot

There are many more fields which get returned on the Process page. These are just the first few. What is important to note is that included in the return is an Id which is the Id of the new Lead that has been created.

Again, it is probably the case that for applications that implement Simpli the page would perhaps redirect to the newly created Lead. This example page just shows all fields for the newly created Lead.

Issues and Exceptions

Following from the above example, if we were to enter invalid data, or miss entering a value we will not receive data as we expect but an error response. Try submitting another Lead but omit the first name.



Processing result (note that below we are displaying all output attributes)

Object Name	Field Name	External Name	Data Type	Value
SYSTEM: Exception	SimpliResultGroupKey		String	
SYSTEM: Exception	SimpliCreateResult		Boolean	false
SYSTEM: Exception	SimpliClassName		String	SimpliFrameworkError
SYSTEM: Exception	SimpliDescription		String	System exception object which is returned when an error occurs during object execution
SYSTEM: Exception	ExceptionCode		String	1
SYSTEM: Exception	ExceptionMessage		String	The following required attributes are not set - First Name
SYSTEM: Exception	ExceptionProcessName		String	Example - Lead Insert
SYSTEM: Exception	ExceptionSystemMessage		String	
SYSTEM: Exception	ExceptionStackTrace		String	
SYSTEM: Exception	ExceptionClassName		String	Simpli.SimpliFrameworkObject
SYSTEM: Exception	ExceptionTimestamp		Datetime	

Figure 4 - Processing Failure Example Screenshot

If there is an error or exception when using Simpli this is the response that will **always** be returned. Some fields may be blank, but they will always be returned explaining the issue that was found.

See Appendix A for detailed information on the above fields.

Parent-Child Configurations

Now that configuration for a single object has been discussed we can move on to configuring multiple objects that are related to each other. One of the more powerful aspects of Simpli is the ability to have the framework perform processing on an arbitrary number of configured objects. For example, we could configure an opportunity and include 2 line items to be inserted, or insert a case and update its associated contact. There are an unlimited number of scenarios that can be configured.

Basic Parent-Child Example – Account And Contact Insert

The example that we will use here is ***Example – Account And Contact Insert (Account)***. This configuration is the top-level configuration.

Field Name	Field Value
Name	Example – Account And Contact Insert (Account)
Class Name	SimpliProcessSFDCObject
SFDC Object API Name	Account
Is Active	true

Table 5 - Example - Account And Contact Insert Config

The top level configuration will always be processed before its children. In this example the attributes are very similar to the above ***Example – Lead Insert*** example and so will not be discussed here. But, this Simpli object has a child configuration associated with it – ***Example – Account And Contact Insert (Contact)***.

Field Name	Field Value
Name	Example – Account And Contact Insert (Contact)
Class Name	SimpliProcessSFDCObject
SFDC Object API Name	Contact
Is Active	True
Parent Object	Example – Account And Contact Insert (Account)

Table 6 - Example – Account And Contact Insert (Contact) Config

Attr. Name	External Name	Type	Allow Override	Is Formula	Formula
ActionType		Input	false		
AccountId		Input	false	true	$parent.Id
FirstName	Contact First Name	Input	true		
LastName	Contact Last Name	Input	true		
Email	Contact Email	Input	true		
Phone	Contact Phone	Input	true		
FirstName	Contact First Name	Output	false		
LastName	Contact Last Name	Output	false		

Table 7 - Example – Account And Contact Insert (Contact) Attributes

Advanced Parent-Child Example - Opportunity Insert

Parent-Child Formula Example

Once a parent has been created, we can use any data on the parent when processing its children.

Action Types

When configuring Simpli for use with standard or custom SFDC objects there are several action types that are available to be used based on configuration needs. Note that it may be the case that a configuration uses multiple action types for the configuration. i.e. Find an account and then use those results to populate and insert an associated object

Insert

This action type is used for the insertion of a record. Typically, values are set based on parent information, configured information which has been defaulted and values set by the user. Once the record has been inserted all fields for that record are retrieved and populated into the Simpli object which is returned.

If using Apex when processing this configuration an SObject can also be passed in to the SFDCObject attribute. Instead of a new record being created this SObject will be inserted. Note that if the SObject has an existing Id the insertion will fail.

Find

This action type is used for finding records. There are a few ways in which to find a record.

Record Id

In its most simple form an Id can be provided to find the necessary record. This method would typically only be used if the configuration will not be presented to the user. An example might be where a record needs to be found so that its children can have processing performed on them.

An example configuration using this method is ***Example – Case Find (Using Id)***

Single Field Key

Using this configuration will probably be the more common method. It requires the user enter a value which is then used to compare against the configured object field. An example configuration using this method is ***Example – Account Find***. The field of importance is configured as follows –

Field Name	Field Value
Name	Id
External Name	Account Name
Allow Overrides	true
Is Required	true
Type	Input
Is Lookup	true
Lookup Field	Account.Name
Lookup Return Field	Id

Table 8 - Single Field Key Attribute Example

In this attribute configuration the Account.Name field has been specified as the primary key to find the record. The value that is provided will be compared against the Account.Name. The record that is returned will have its Id field set into the value.

Note that the default find action checks for queries that return multiple rows. If multiple rows are returned the process will fail. This can be changed for the individual configuration by setting the attribute **SimpliSOQLGetFirst** to true. In this case if there are multiple records that match the criteria the first record will be returned.

Composite Field Key

In some cases there will be multiple fields that need to be used to find a record. In these cases the field values need to be separated by designated set of characters. i.e. !!

An example of the usage for a composite key is in config with name ***Example – Opportunity Line Items Insert*.** When inserting a line item a price book entry must be provided. But, to determine the entry both a product name and a price book must be provided in order to find the entry. To accomplish this an attribute is created which retrieves the Id of the price book entry based on two values. The attribute is configured as follows –

Field Name	Field Value
Name	PriceBookEntryId
External Name	Opportunity Line Item Product Name
Allow Overrides	true
Is Required	true
Type	Input
Is Lookup	true
Lookup Field	Product2.Name!!PriceBook2.Name
Lookup Return Field	Id

Table 9 - Composite Key Attribute Example

The main thing to notice here is the lookup field. What this is saying is the following –

1. Take the value supplied and use it to find the Product Id using the Product2.Name as they lookup key.
2. Take the value from the attribute called PriceBook2.Name which provides the Id of the pricebook.
3. Find the object which is a lookup to PriceBookEntryId (which in this case is PriceBook) and use the lookup attribute names and values to find the pricebook entry.
4. Take the Id of the pricebook entry that was found and put it into this attributes value to be used when inserting the opportunity line item.

Initially this will seem daunting. After using this style for a while it will begin to make sense.

Update

The update action type is used to update a record. This typically means the record needs to be found based on an Id. The config with name ***Example – Contact Update*** gives a basic idea of how this would be configured.

Specifically, the attribute which determines the Id of the contact is configured as follows –

Field Name	Field Value
Name	Id
External Name	Contact Id
Allow Overrides	true
Is Required	true
Type	Input

Table 10 - Update Simple Example

In cases where external systems need to update SFDC records this might make sense as the Id could be available to use. But, in other cases this configuration might not be appropriate as the Id of the record might not be available. In this situation the attribute could be configured to allow the user to enter more user-friendly information.

Field Name	Field Value
Name	Id

External Name	Contact Name
Allow Overrides	true
Is Required	true
Type	Input
Is Lookup	true
Lookup Field	Contact.Name
Lookup Return Field	Id

Table 11 - Update Lookup Example

In this example the contact name is passed into the config and its used to find the contact to update.

Delete

The delete action type is configured in the same way as the find and update action types above. For the record to be deleted it needs to be found. The record can be found using the basic Id, simple lookup or more complex composite key lookup.

See above examples for attribute configuration as well as the ***Example – Account Delete*** configuration.

Clone

The clone action type is used when needing to duplicate records. Configurations can have multiple levels of referenced objects which are cloned. Therefore, whole graphs of objects can be cloned. Very useful for example when needing to create test data.

Simple Clone

From a simple perspective records can be cloned on an individual basis. An example configuration for cloning a single record is ***Example – Object Clone (Account)***. When cloning objects attributes are used to handle any fields that should be updated before the object is inserted. For the above example all fields on the account remain the same except for the Name field which is provided by the user.

Below is a screenshot giving the basic configuration along with the attributes.

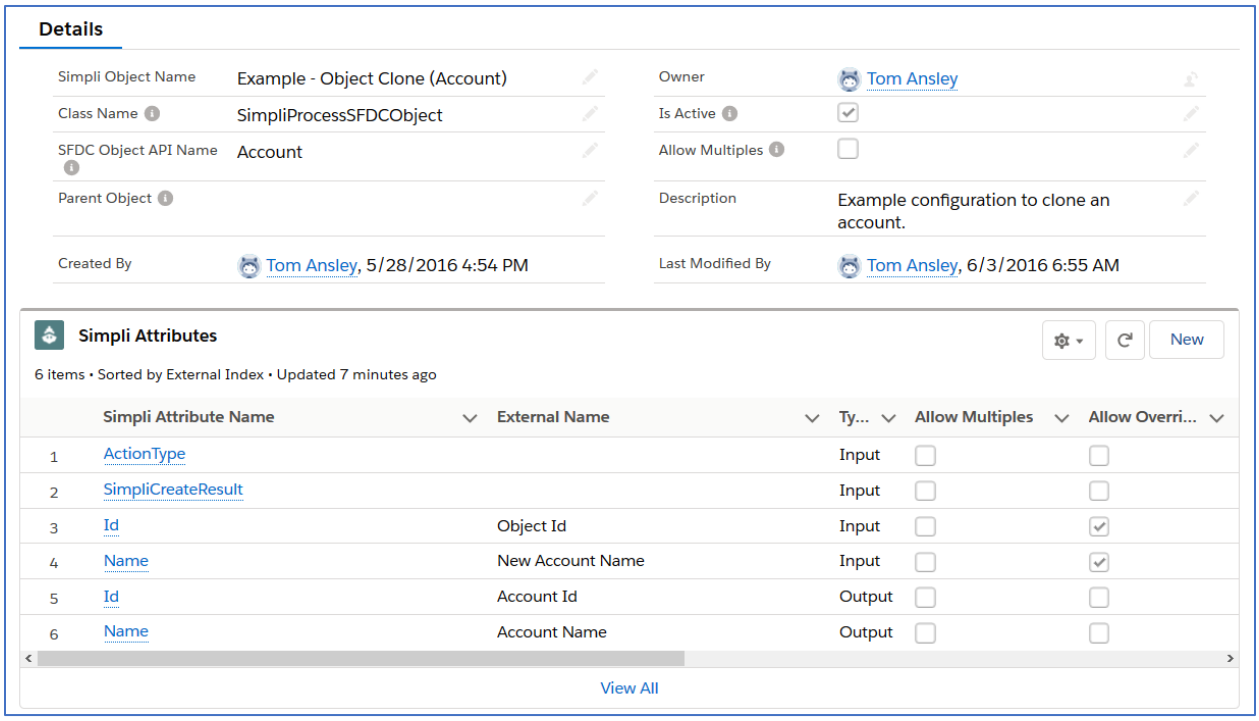


Figure 5 - Clone Account Example Screenshot

Advanced Cloning and Handling Unique Values

Cloning single records is useful. But cloning whole graphs of data can save a lot of time when needing to create records for any number of tasks. One of the topics covered further down the document, the handling of indeterminate numbers of children objects, allows cloning when any number of children of the parent object need to be handled. Please read that section if advanced cloning is required.

An example configuration which handles all the topics discussed in this section is the ***Example – Object Clone (Simpli Object)*** config. This configuration uses Simpli to clone its own Simpli configurations! The graph of the objects that are cloned is as follows.

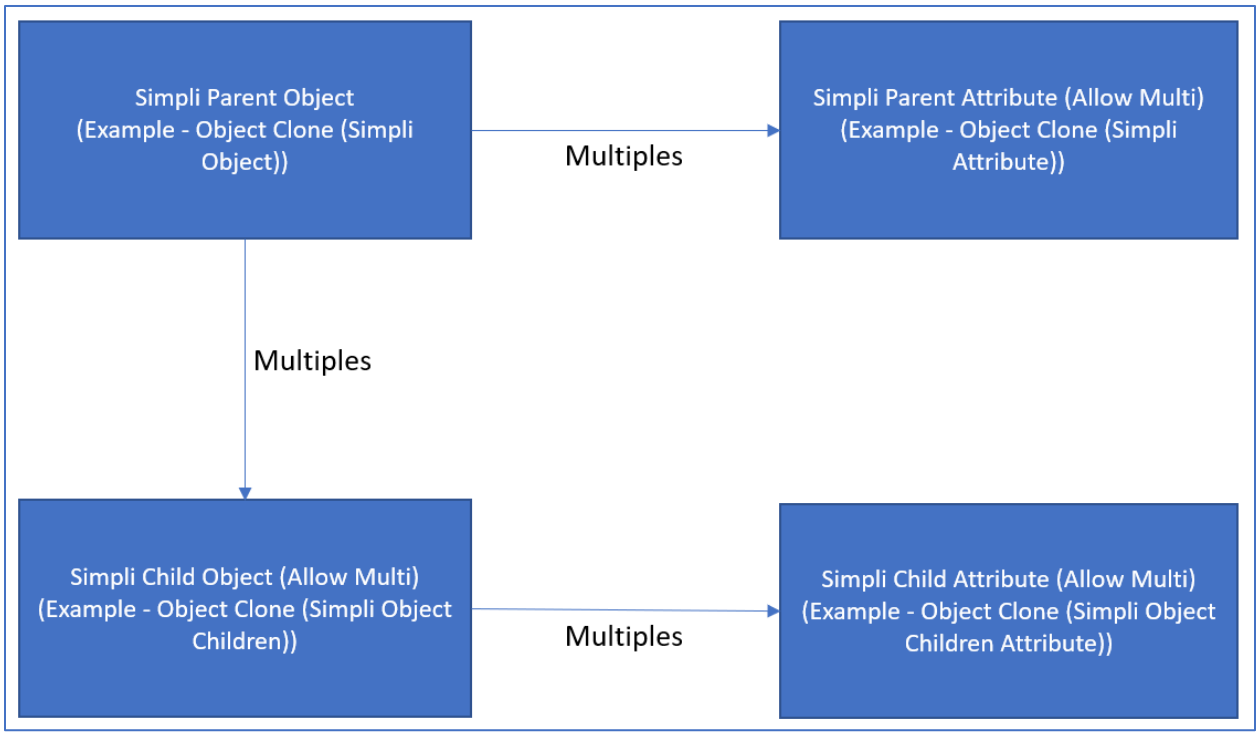


Figure 6 - Simpli Object Clone Entity Relationship Diagram

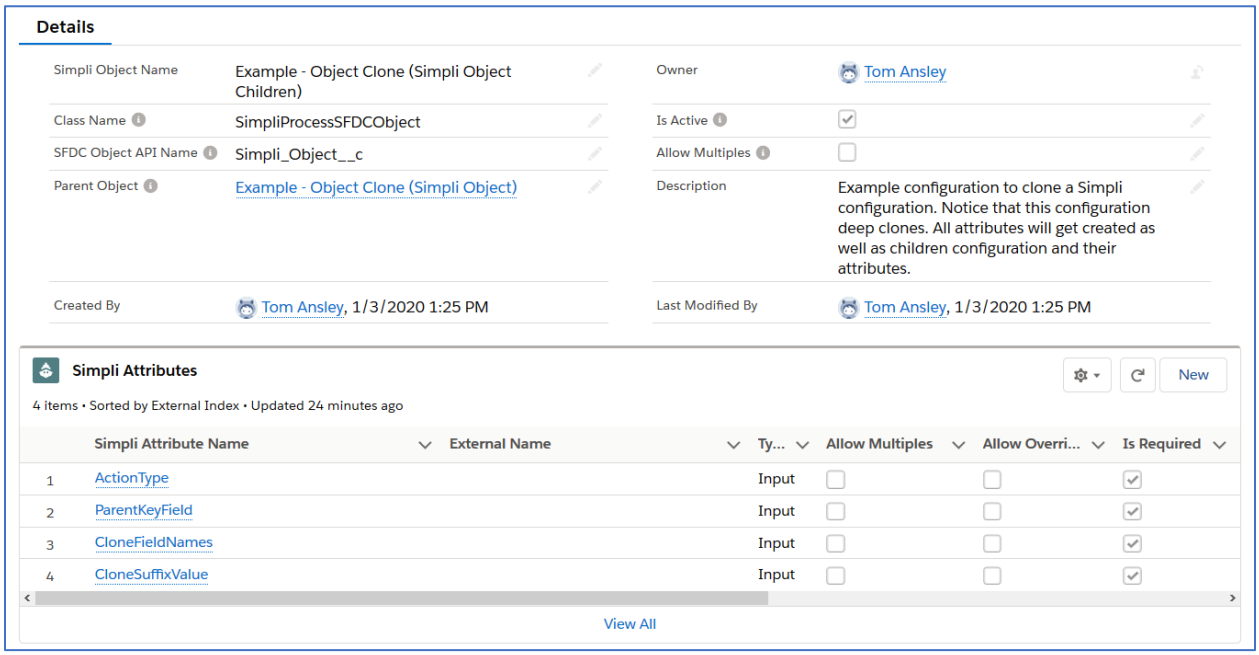
A few things to note –

- We are using Allow Multiples for 3 of the object configurations. Not only that, we have, depending on the configuration being processed, multiple child objects with multiple child attributes!
- In this use-case we do not want the names of the newly cloned configurations to identical to those that already exist as processing would fail due to validation rules.
- This configuration only goes two Simpli objects deep. If deeper cloning is required changes would be needed on the existing configuration. If deeper cloning is required, please keep governor limits in mind as objects could grow exponentially!

In many cases when cloning records new values need to be set to handle fields that are required to be unique. This becomes tricky when handling indeterminate numbers of child objects. The way this is handled is as follows –

1. supply the API names of those fields that need to be updated by setting the attribute with name **CloneFieldNames**. Multiple fields can be set using the !! delimiter.
2. Indicate whether we are to prefix the existing value or postfix it by setting the attribute with name **ClonePrePostSuffix** to either “Pre” or “Post”. If no value is set then by default the field is set to “Post”.
3. Indicate the suffix that is to be set on to the unique fields by setting the attribute with name **CloneSuffixValue**. If no value is set then by default the field is set to <Existing Value>-currentTimeInMillis.

Note that currently the unique fields must be able to be set using a string value. An example config which handles the unique naming of fields is ***Example - Object Clone (Simpli Object Children)***. The config and its attributes would look as follows.



And the 2 clone specific attributes would be configured as follows –

Field Name	Field Value
Name	CloneFieldNames
Is Required	true
Type	Input
Value	Name

Field Name	Field Value
Name	CloneSuffixValue
Is Required	true
Type	Input
Is Formula	true
Formula	$parent.Name

In this case we are setting the suffix of the fields to the parent objects name.

Publish

The publish action type is used for platform event publishing. For more information on setting up platform event objects and fields look at the following link - https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_publish.htm. This document will not discuss the setting up and nuances of platform events.

Once the platform event object has been set up from a Simpli perspective it is handled in a very similar way than any other object. An example of using the publish action type is ***Example – Platform Event Insert***.

Field Name	Field Value
Name	Example - Platform Event Insert
Class Name	SimpliProcessSFDCObject
SFDC Object API Name	Simpli_Platform_Event__e
Is Active	true
Parent Object	

Table 12 - Example Publish Action Example

Note that the class name for the example config uses **SimpliProcessSFDCObject**. The same as other SFDC objects. Also, In order to test this example a platform event type with API name Simpli_Platform_Event__e must be created along with a field with API name Simpli_Text_Field__c.

Formulas

Formulas are used to retrieve values from a different location. There are currently three formula labels that can be used when retrieving values. The general format for a formula is as follows –

```
$LABEL.ATTRIBUTE_NAME
```

- $label - identifies what object or location the data is being retrieved from.
- ATTRIBUTE_NAME – the attribute name of the field on the object or location that holds the data.

An example is as follows - $parent.Name. This formula is saying to go to the parent object of this configured object and get the value in the attribute called Name. The retrieved data will then be used as the value for this configured attribute.

$this

The $this label identifies that the data should be retrieved from the same object as the object the formula attribute is configured for. An example which uses $this is ***Example – User Insert***. The attribute is configured as follows.

Field Name	Field Value
Name	CommunityNickname
External Name	
Allow Overrides	false
Is Required	true
Type	Input
Is Formula	true
Formula	$this.Alias

Table 13 - $this Example Attribute

This attribute is saying to take the value that is in the Alias attribute of the same object and use its value.

$parent

The $parent label identifies that the data should be retrieved from the objects parent object. For example, with config ***Example – Account And Contact Insert (Contact)*** the object has a parent called ***Example – Account And Contact Insert (Account)***. We could use $parent to retrieve the accounts Id attribute value from the parent account to use on the child contact to create their relationship. The attribute would look like the following.

Field Name	Field Value
Name	AccountId
External Name	
Allow Overrides	false
Is Required	true
Type	Input

Is Formula	true
Formula	$parent.Id

Table 14 - $parent Example Attribute

$user

The $user label identifies that the data should be retrieved from the currently running user. For example, with config ***Example - Task On Account Insert*** we are setting the owner of the task to be the running user.

Field Name	Field Value
Name	OwnerId
External Name	
Allow Overrides	false
Is Required	true
Type	Input
Is Formula	true
Formula	$user.Id

Table 15- $user Example Attribute

Lookups

Lookup attributes are used to retrieve data from an object other than itself or parent object. These attributes have already been seen in a number of examples in earlier sections. Examples of using a lookup is when needing to find a record to update it, or find an Id to be used when saving another record. An example of this for finding a record might be the ***Example - Case Create - Electrical Issue*** where the name of an account is used to lookup the account Id which is used on the case.

Field Name	Field Value
Name	Id
External Name	Account Name
Allow Overrides	True
Is Required	True
Type	Input
Is Lookup	True
Lookup Field	Account.Name
Lookup Return Field	Id

Table 16 - Basic Lookup Example

In the above example it is expected that the Account name will be provided in order to find the account. A more complicated example is where an attribute value needs to be used for the lookup rather than a provided value. An example of this is on the Example – Case Create (Email) config.

Field Name	Field Value
Name	ReplyTo
External Name	Account Name
Type	Input
Is Lookup	true
Lookup Field	User.Id
Lookup Return Field	Email
Lookup Id Value	$parent.OwnerId

Table 17- Advanced Lookup Example

Here, a value from the objects parent needs to be used. In this case the owner Id is retrieved from the parents OwnerId attribute. This owner Id is then used to find a user with the same Id. The Email field on the returned user is then used as the value to be used on the ReplyTo attribute.

As above for formula's this can be daunting when first configuring but makes sense as familiarity sets in.

Web Services

Simpli was developed more for web services vs. being used internally on the SFDC UI. Salesforce makes it easy to create records and configure which fields are displayed and how for individual objects. But this is at a table level and so is table driven.

Simpli, works at a more abstract level, allowing for the configuration and processing of graphs of objects. i.e. an object can be processed, then it's children can be processed and then those objects children can be processed, allowing us to configure processing based on use-cases, rather than tables.

From an interface perspective the web service works in an identical way to internally using Apex. i.e. the process name and all its externally named attribute values must be supplied.

REST

This section assumes the reader has familiarity with using REST services on the SFDC platform. This includes the oAuth security model which is typically how the REST services are secured. Once the oAuth token has been retrieved based on the user credentials and Connected App credentials the REST service request needs to include the following information –

Field Name	Field Value
Endpoint	https://<SALESFORCE-ORG>/services/apexrest/Simpli/Simpli/
Parameter 1 (Authorization)	Bearer <TOKEN>
Parameter 2 (Content-Type)	application/json
Body	<JSON-SIMPLI-REQUEST>

Table 18 - Example REST Request Settings

Examples of the above are as follows –

- Endpoint - https://simpli-interface-dev.na69.force.com/services/apexrest/Simpli/Simpli/
- Parameter 1 – Bearer 00D15000000FSzl!AQYAQKPlXipoPePGQsGKgyOKftByWfHOrsd
- Body

```
{
        "requestData" :
        {
                "processName":"Example - Account Insert",
                "fields":[
                        {"fieldValue":"Test Account Name 12534",
                         "fieldName":"Account Name"},
                        {"fieldValue":"123",
                         "fieldName":"Number of Employees"},
                        {"fieldValue":"10101",
                         "fieldName":"Zip"}
                ]
        }
}
```

No matter how complicated the configured process the construct will hardly change regardless of the number of objects being created or the depth of the hierarchy. How the processes are configured might be complex, but the interface will never be that different.

Below is an example of the most complex the interface body construct gets! This example creates an opportunity and an indeterminate number of opportunity line items (in this example 2). Hence the need for an index to identify which piece of data goes with which line item.

```
{
    "requestData" :
    {
        "processName":"Example - Opportunity Insert (Multi)",
        "fields":[
            {"fieldName":"Opportunity Close Date", "fieldValue":"11/20/2019"},
            {"fieldName":"Opportunity Name", "fieldValue":"Opp Test 12348"},
            {"fieldName":"Opportunity Order Number", "fieldValue":"123214"},
            {"fieldName":"Opportunity Stage", "fieldValue":"Prospecting"},
            {"fieldName":"Opportunity Line Item Product", "fieldValue":"Product 1", "fieldIndex":"0"},
            {"fieldName":"Opportunity Line Item Quantity", "fieldValue":"1", "fieldIndex":"0"},
            {"fieldName":"Opportunity Line Item Total Price", "fieldValue":"100", "fieldIndex":"0"},
            {"fieldName":"Opportunity Line Item Product", "fieldValue":"Product 2", "fieldIndex":"1"},
            {"fieldName":"Opportunity Line Item Quantity", "fieldValue":"2", "fieldIndex":"1"},
            {"fieldName":"Opportunity Line Item Total Price", "fieldValue":"400", "fieldIndex":"1"}
        ]
    }
}
```

Notice how, based on the provided "field names" it can be determined for which object the data is intended. The configured processes could get even more complicated, but the interface will never get more complicated than the above.

SOAP

This section to be completed.

Batch

The batch capability of Simpli allows for the running of a process multiple times against multiple records that have been preselected. The intention here is to allow a user interface to list multiple records which get selected and processed.

NOTE – batch is not designed to be used for large-scale loading of data.

This section to still be completed! Please reach out with any questions until then.

Advanced Functionality

What follows are more advanced capabilities that may be useful during configuration of a Simpli object process.

Processing Order of Execution (SimpliFrameworkSFDCObject)

The order of execution after starting the processing of a Simpli object can be important when handling formulas, lookups and child objects. The order is as follows –

1. Object has its formula fields evaluated and set.
2. Object is validated.
3. Object lookup fields are evaluated and set.
4. Object is processed based on action type.
5. Object output attributes are synced with the latest associated SFDC record.
6. Object children are processed
    a. If child object has a parent key field, then it is evaluated and set.
    b. If child object is being cloned, then expand out parents' children for cloning.
    c. Child object is processed. (repeat from step 1 until complete)
7. If Object has a "NextProcess" attribute configured –
    a. Create "NextProcess" Simpli object.
    b. Retrieve all output external attributes from original process and set into the new process.
    c. "NextProcess" object is processed (repeat from step 1)

Note the following –

- Lookup fields can use formula field values.
-

Daisy Chaining Processes

It may be the case that once a process has completed running another process needs to be run. It could also be the case that configurations are broken into small reusable configurations and those configurations need to be processed sequentially. In these cases it is possible to configure processes such that they call the next process immediately.

To configure a process to run after another process has finished the configured process must include an attribute called **SimpliNextProcess** with the name of the process to run as the value. For example, the following attribute will ensure the process with name Example – Event Insert gets called and processed.

Field Name	Field Value
Name	SimpliNextProcess
Type	Input
Value	Example – Event Insert

Please note the following regarding chaining processes.

- All processes are run in the same transaction to ensure data integrity. Ensure governor limits are handled appropriately.
- The calling process must have output attributes which align from an external name perspective with the input attributes required by the called process.
- Only the result of the last called process will be returned, unless the process has been configured to be output to the Simpli result object.
- The attribute values from the calling process are made available to be used by the called process.

Handling Indeterminate Number of Child Objects

Simpli can process configurations where an indeterminate number of children need to be updated or created. This is common for example when creating a configuration to create an opportunity with line items where the line items may or may not be needed. An example configuration which uses this capability is the ***Example – Opportunity Line Items Insert (Multi)***. The configuration for the object is as follows –

Field Name	Field Value
Name	Example – Opportunity Line Items Insert (Multi)
Class Name	SimpliProcessSFDCObject
SFDC Object API Name	OpportunityLineItem
Is Active	true
Parent Object	Example – Opportunity Insert (Multi)
Allow Multiples	true

Table 19 - Indeterminate Number of Child Objects Example

The thing to notice here is that "Allow Multiples" is set to true. Note that only configured processes with parent objects can allow multiples.

Now that the object allows multiples we can create a process where multiple children are created. Note that for each child created we need to supply all required externally named attribute values. Here is an example REST request which uses the ***Example – Opportunity Insert (Multi)*** config to create an opportunity with multiple line item records (even though only one line item record has been configured).

```
{
    "requestData" :
    {
        "processName":"Example - Opportunity Insert (Multi)",
        "fields":[
            {"fieldName":"Opportunity Close Date", "fieldValue":"11/20/2019"},
            {"fieldName":"Opportunity Name", "fieldValue":"Opp Test 12348"},
            {"fieldName":"Opportunity Order Number", "fieldValue":"123214"},
            {"fieldName":"Opportunity Stage", "fieldValue":"Prospecting"},

            {"fieldName":"Opportunity Line Item Product", "fieldValue":"Product 1", "fieldIndex":"0"},
            {"fieldName":"Opportunity Line Item Quantity", "fieldValue":"1", "fieldIndex":"0"},
            {"fieldName":"Opportunity Line Item Total Price", "fieldValue":"100", "fieldIndex":"0"},

            {"fieldName":"Opportunity Line Item Product", "fieldValue":"Product 2", "fieldIndex":"1"},
            {"fieldName":"Opportunity Line Item Quantity", "fieldValue":"2", "fieldIndex":"1"},
            {"fieldName":"Opportunity Line Item Total Price", "fieldValue":"400", "fieldIndex":"1"}
        ]
    }
}
```

Salesforce Permission Handling

From a high level Simpli uses the running users' profile to determine permissions. Simpli will not allow a user to create, access or update a record that they do not have permission for. During transaction processing, object metadata is used to determine eligibility. This means that the standard OOTB permissions are used. This also applies to custom objects and fields. As soon as a field is added it can be used by Simpli. The security permissions for that field will be applied.

Please note that if users do not have field access Simpli will error with a message displaying the issue. It may in some cases not be able to determine permissions. In these cases, Simpli may perform in an erratic manner depending on the use case.

Appendix A (Table of Attribute Fields Per Object Type)

SimpliProcessSFDCObject

Attr. Name	Required	Allow Override	Description
SFDCObjectName	true	true	Indicates the SFDC SObject type that this object is associated with. The SObject is used to generate available fields on the Simpli Object
SFDCObject	false	true	Used to hold a concrete SObject if retrieved.
ActionType	true	true	Indicates the action to be performed by the Simpli object. For this object the possible values are Find, Insert, Update, Delete, Clone or Publish.
ParentKeyField	false	true	If the Simpli object is a child this indicates the field on the parent that holds the parents lookup Id. This is used for SFDC lookups.
FieldName	false	true	Holds the name of a field that will be used when performing Find, Update, Delete or Clone actions.
FieldValue	false	true	Holds the value of the field that will be used when performing Find, Update, Delete or Clone actions.
CloneSuffixValue	false	true	Holds the suffix used when cloning objects. This helps when cloning objects and child objects that must have unique names.
ClonePrePostSuffix	false	true	Holds whether the pre/suffix value is for a prefix or a postfix when cloning objects. This helps when cloning objects and child objects that must have unique names.
CloneFieldNames	false	true	Holds the Simpli field names which must be altered using the pre/post suffix value when cloning objects. This helps when cloning objects and child objects that must have unique names.
			Note that for all SimpliProcessSFDCObject configurations when creating the object the framework will build attributes for all fields that exist on the provided SFDCObject dynamically. This means that attributes using SFDC object fields can also be configured.
SFDCPlatformEventResult	false	false	Holds whether the platform event was successfully published (if this object is a platform event)

SFDCPlatformEventErrorDesc	false	false	Holds the error description if the platform event was not successfully published (if this object is a platform event)

Table 20 – SimpliProcessSFDCObject

SimpliProcessSFDCObjectBatch

Attr. Name	Required	Allow Override	Description
SFDCObjectName			
ActionType			Indicates the action to be performed by the Simpli object. The possible values are Update, Truncate or Export.
FieldName			Holds the field name or !! separated field names which will be exported or updated.
FieldValue			Holds the field values or !! separated values which will be used when updating.
ExportLocationType			Indicates where the exported data should be placed. Currently the only allowed value is Attachment.
ExportFilename			Indicates the filename prefix of the export attachment. A date timestamp and ".csv" are always appended to the name. For example, if the filename is set to "Test" the result would be "Test-2020-10-23-09-10-45.csv". This ensures files do not overwrite each other. Future capabilities will include the ability to indicate the suffix, or none at all.
ExportParentId			Indicates the record that the attachment should be associated with. Note that currently files are created as Attachments. Future capabilities will allow Files to be created and associated.

SimpliProcessEmail

Attr. Name	Required	Allow Override	Description
EmailTemplateName	false	true	Holds the name of the SFDC email template that is to be used when constructing the email. Either the template name must be supplied or an email body provided

OrgWideEmail	false	true	If the email is going to an org wide email then that email address is placed in this field.
ReplyTo	false	true	The reply-to email address of the email.
ToEmailAddress	false	true	The email address that the email is going to. If this email has a target object type then this field must be populated before processing.
TargetObjectType	false	true	Holds the target type of the email address. This can currently be one of - Contact, Lead and User.
WhatId	false	true	The WhatId value which is used by the email template provided.
EmailBodyType	false	true	Indicates the type of the email body. This can be one of - HTML or Plain
EmailBody	false	true	The body of the email if a template is not being used.
EmailSubject	false	true	The subject of the email.

SimpliProcessSimpliConfig

Attr. Name	Required	Allow Override	Description
ActionType	true	true	Indicates the action to be performed on the Simpli configuration. The possible values are Find and Clone.
ConfigName	true	true	Indicates the name of the configuration that is being handled.

SimpliFrameworkError

Attr. Name	Required	Allow Override	Description
ExceptionCode	true	false	A unique error code
ExceptionMessage	true	false	A user-friendly error message
ExceptionProcessName	true	false	The configuration that was being processed at the time of the error.
ExceptionStackTrace	false	false	The stack trace if an exception was thrown. Can be used for debugging purposes.
ExceptionSystemMessage	false	false	The message provided by the exception if an exception was thrown
ExceptionClassName	False	false	The exception class name if an exception was thrown.
ExceptionTimestamp	true	false	Timestamp when the error occurred.

Table 21 - SimpliFrameworkError